

Linking a SAP Build app to Microsoft Teams

I have been investigating easy ways to demonstrate the capabilities of Loopyl, the SAP:Microsoft Teams integration platform.

One great feature of Loopyl is the lo-code / no-code approach to building Teams integrations, which makes it a fantastic tool for citizen developers.

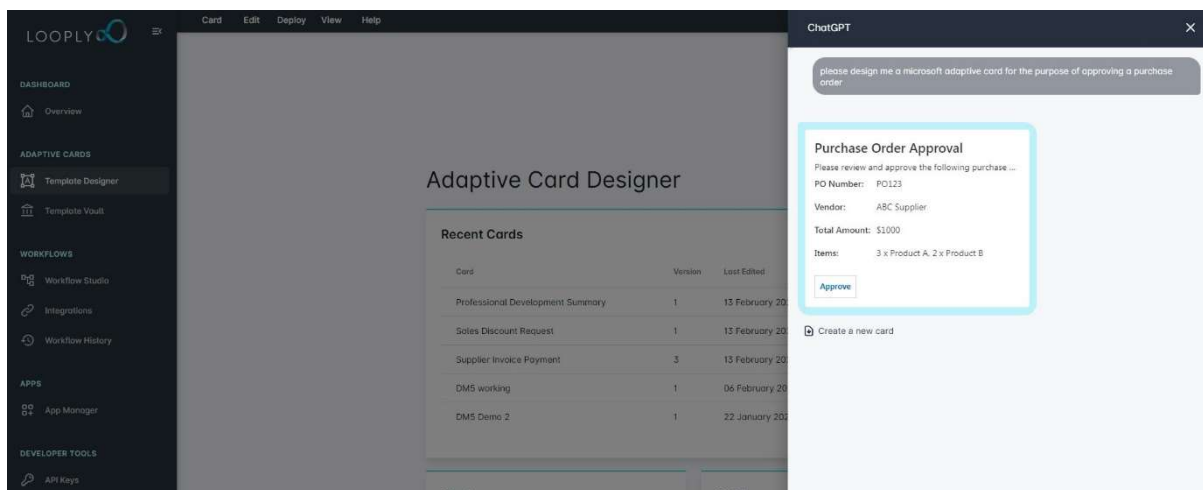
So, since SAP Build is the cornerstone of SAP's citizen developer offering, I decided to try to build a simple app using SAP Build, and see what steps are required for the app to trigger the creation of adaptive cards within Teams.

Like everything, as I was starting with zero knowledge of SAP Build and little knowledge of Loopyl, the first time involved a lot of learning. But with a little experience of both, the integration turns out to be ridiculously simple.

Building demo cards

I didn't set off with a specific business process to model, but instead decided to build a series of cards with hard-coded data to simply illustrate what could be possible.

So without any back-end process to worry about, I could immediately start building some sample cards within Loopyl.



The option to use AI to build a card proved very useful. Basically there's an in-built integration with ChatGPT that enables you to define your requirements and a card is generated for you.

I could then edit the generated card using the Template Designer, and then create a copy, change all the fields and re-save.

Linking a SAP Build app to Microsoft Teams

The screenshot displays the SAP Build Studio interface. On the left, there are two panels: 'CARD ELEMENTS' and 'Inputs'. The 'CARD ELEMENTS' panel is divided into 'Containers' (Container, ImageSet, FactSet, ColumnSet, Table) and 'Elements' (TextBlock, RichTextBlock, Image, Media, ActionSet). The 'Inputs' panel lists various input types like Input.Text, Input.Date, Input.Time, Input.Number, Input.ChoiceSet, and Input.Toggle. The central workspace shows a 'Leave Request' card with the following content: '3.00 days', 'Chris Scott', 'Employee ID 12345', 'Annual Leave', 'Monday 25 November 2024 - Wednesday 27 November 2024', a table with 'Available Balance 24 days', 'Requested 3 days', 'Total Deduction 3 days', and 'Leave Type Annual Leave', 'Notes' (Chris Scott 24 February 2023, 10:05: Extended weekend break), an 'Add note (optional)' text input, and 'Approve' and 'Reject' buttons. On the right, the 'CARD STRUCTURE' panel shows a tree view of the card's components, including AdaptiveCard, Container, ColumnSet, Column, TextBlock, FactSet, RichTextBlock, Input.Text, and Action.Submit.

In this way I was able to create a few different example cards very quickly, as of course there was no data binding or dynamic logic to consider.

Building the Loopy logic

The Loopy logic takes the form of a Workflow, which is built using the Workflow Studio.

The Workflow needs to be built alongside the SAP Build app. While not necessarily an iterative process, the key steps need to be:

- 1) Begin the Loopy Workflow build, and save to create the API 'endpoint'
- 2) Build the Loopy app, and trigger the Loopy Workflow from the app in test mode, to pass in the data model
- 3) Complete the build of the Loopy Workflow, now with visibility of the data model.

Starting the Workflow build means literally giving the workflow a name, selecting the 'event trigger' starting point, and hitting Save.

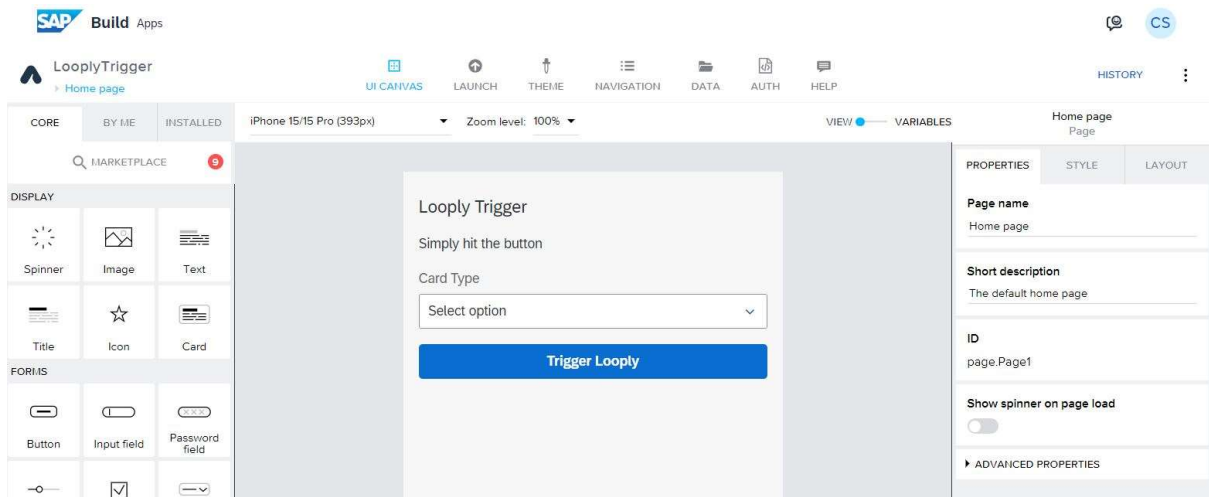
The screenshot shows the 'Trigger Loopy demo' interface in the Workflow Studio. It features a central workspace with a grid background and a single 'Event Trigger Request' block. To the right, there is a configuration panel with a 'Save' button at the top. Below it, the 'Edit Event Trigger' section includes a 'Select an event trigger' dropdown menu set to 'Request'. The 'Request URL' section shows a POST request to 'https://api.loopy.io/v2/workflows/triggerworkflow' with a 'Copy' button. The 'Payload' section is partially visible at the bottom.

This generates a 'Request URL' that you can copy, as you will need this in your SAP Build app.

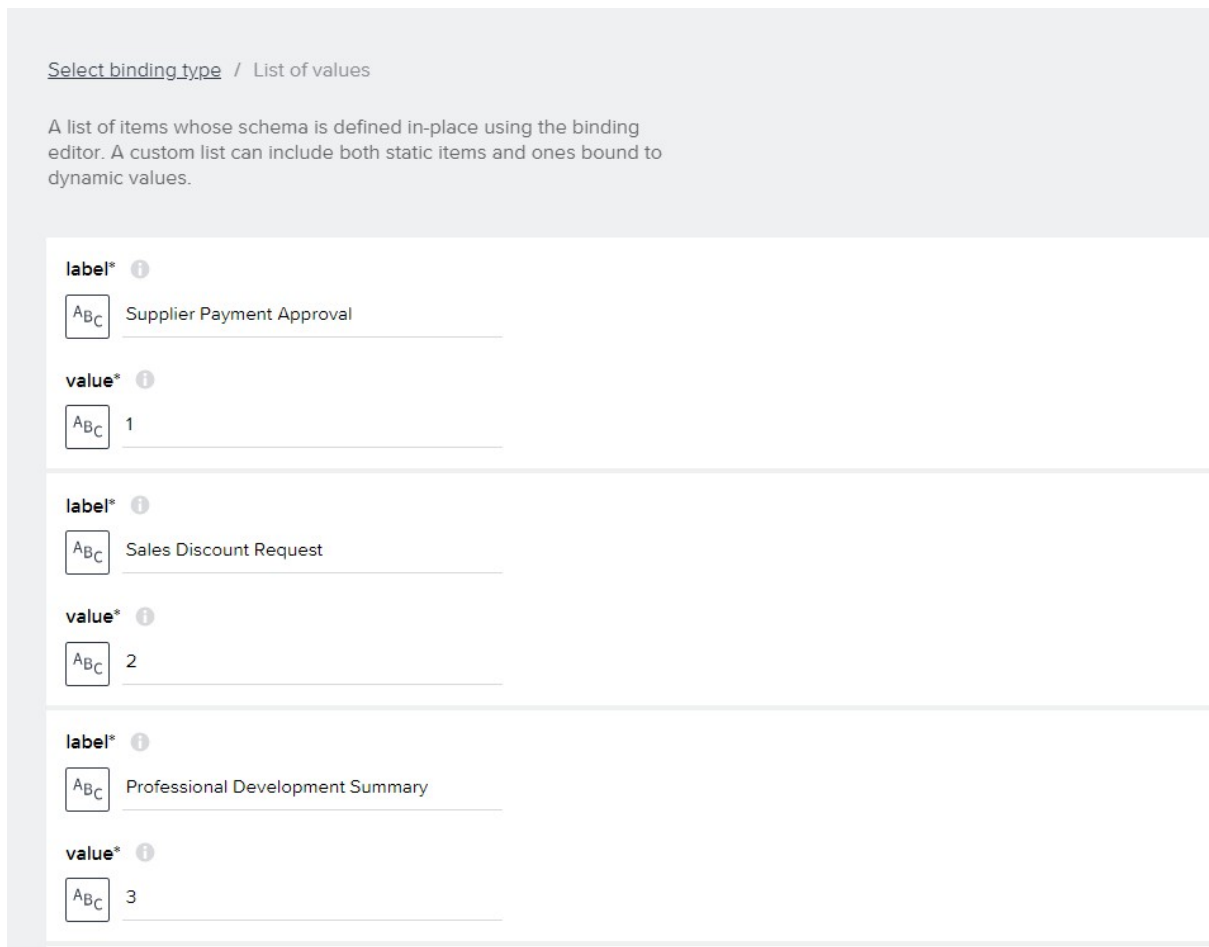
Linking a SAP Build app to Microsoft Teams

Building the SAP Build app

For the SAP Build app design I added a heading, subheading, a drop-down list and a button.

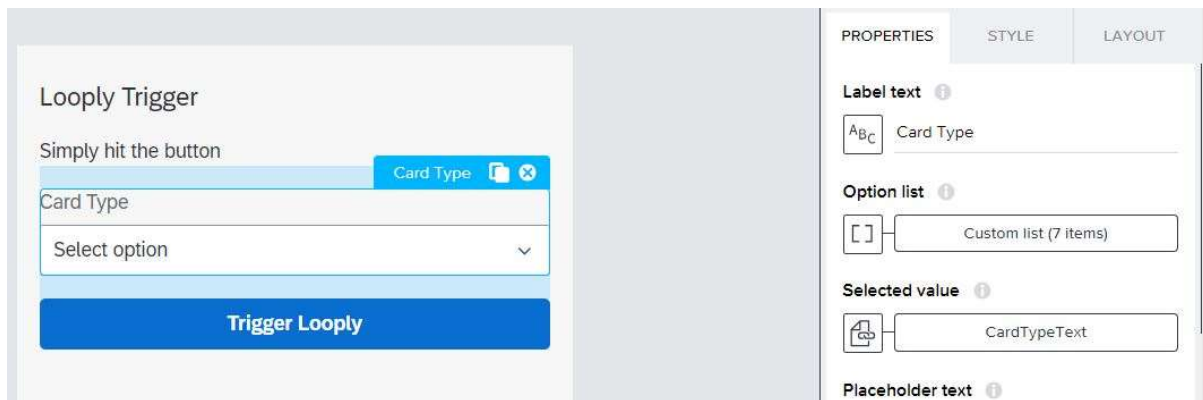


I hardcoded values in the drop-down list, giving each entry a integer value and a readable label that related to the sample cards I built in Loopy.

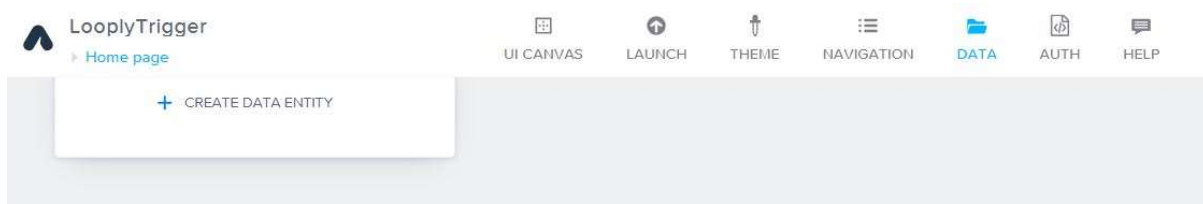


Linking a SAP Build app to Microsoft Teams

I then created a 'Page variable' called 'CardTypeText' and bound the Selected Value of the drop-down list to the variable.



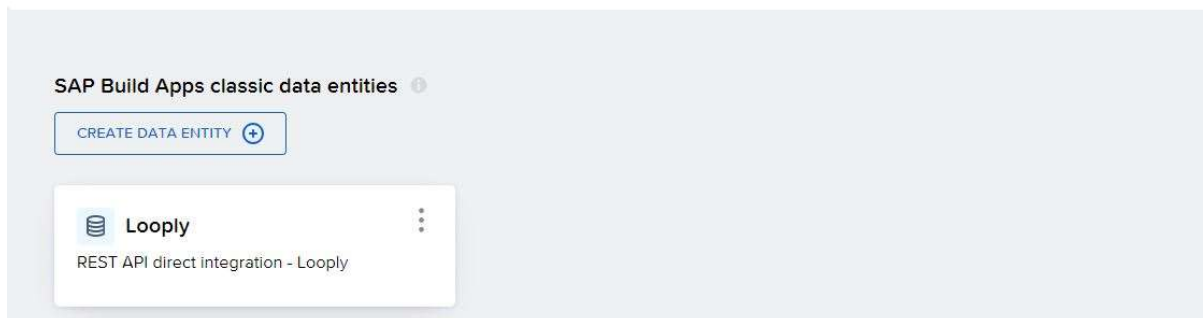
Next I needed to define my data connection to Looply.



No systems integrated

Integrate your app with your chosen backend to display dynamic, up-to-date content in your app.

[ADD INTEGRATION](#)



I used the SAP Build Apps classic data entities option, and pasted in the 'Request URL' string from the Looply Workflow into the 'Resource URL' field. I also added an HTTP Header to pass in a generated Looply API key.

Linking a SAP Build app to Microsoft Teams

Configure Direct REST Integration

REST API direct integration resource **Looply**

BASE	Resource ID								
GET COLLECTION (GET)	Looply								
GET RECORD (GET)	Short description								
CREATE RECORD (POST)	Looply								
UPDATE RECORD (PUT)	Resource URL								
DELETE RECORD (DELETE)	https://api.looply.io/v2/workflows/triggerworkflow/d38ea30c-422b-461e-b15.								
	HTTP Header								
	<table border="1"><thead><tr><th>KEY</th><th>LABEL</th><th>VALUE</th><th>DESCRIPTION</th></tr></thead><tbody><tr><td>x-api-key</td><td>x-api-key</td><td>EHZxRYzFMi6jt3UpZzHTL3J4Ch0oi</td><td>(unset)</td></tr></tbody></table>	KEY	LABEL	VALUE	DESCRIPTION	x-api-key	x-api-key	EHZxRYzFMi6jt3UpZzHTL3J4Ch0oi	(unset)
KEY	LABEL	VALUE	DESCRIPTION						
x-api-key	x-api-key	EHZxRYzFMi6jt3UpZzHTL3J4Ch0oi	(unset)						
	URL placeholder								
	No URL placeholder added.								
	Query parameter								
	No Query parameter added.								

SAVE DATA ENTITY

Under 'Get Collection' I added a field 'CardType' to the schema, and then I could save the data entity.

Configure Direct REST Integration

REST API direct integration resource **Looply**

BASE	CONFIG	TEST	SCHEMA	Method enabled
GET COLLECTION (GET)	Get collection (GET) response schema			<input checked="" type="checkbox"/>
GET RECORD (GET)	Use GET schema			
CREATE RECORD (POST)	Schema properties			
UPDATE RECORD (PUT)	Properties of this schema			ADD PROPERTY +
DELETE RECORD (DELETE)	CardType			integer text
	id			text

SAVE DATA ENTITY

Now, back in the UI Canvas, I could add some logic to the button.

Linking a SAP Build app to Microsoft Teams

The screenshot displays the SAP Build IDE interface. At the top, there are navigation tabs: CORE, BY ME, and INSTALLED. Below this is a search bar for the marketplace. The main workspace is divided into two sections: the top section shows the UI canvas for a component named 'Button 1', and the bottom section shows the logic canvas for 'Component: Button_1'. The logic canvas contains a sequence of actions: an event 'Component tap', a variable action 'Set page variable' (with sub-action 'Set CardTypeNum to [variab...]', and a data action 'Create record Loopy'. The right sidebar shows the properties panel for 'Button 1', including fields for Label, Disabled, Repeat with, and Component display name.

Here I passed in the page variable to the data connection schema.

This screenshot shows the 'Record properties' configuration screen. It features a heading 'Select binding type / Object with properties' and a descriptive paragraph: 'An object whose schema is defined in-place using the binding editor. A custom object's properties can contain both static and dynamic bindings.' Below this, there are two rows of configuration: 'Card Type*' and 'ID*', each with a 'CardTypeText' input field.

Back in the data connection, I was ready to use the 'RUN TEST' option under 'CREATE RECORD' to try to trigger the Loopy Workflow.

This screenshot shows the 'Configure Direct REST Integration' screen. It has a heading 'Configure Direct REST Integration' and a sub-heading 'REST API direct integration resource Loopy'. Below this is a table with columns for 'BASE', 'CONFIG', 'TEST', and 'SCHEMA'. The 'TEST' tab is active, showing a 'Record properties*' section with a 'Custom object' input field and a 'Test API call response' section with a text area. A 'RUN TEST' button is visible in the top right corner of the TEST tab.

Linking a SAP Build app to Microsoft Teams

Completing the Loopy Workflow

Back in the Loopy Workflow, I can now add a 'branch' step based on a condition, which checks the value of the 'CardType' field.

Trigger Loopy demo Active Version 8 Last saved 12 days ago Save Delete

Event Trigger Request

Branch Conditional My Condition

Supplier Payment

Subcontract PO

Purchase Requisition

Outgoing Claim

Adaptive Card Action My Action

Adaptive Card Action My Action

Adaptive Card Action My Action

Adaptive Card Action My Action

Name: My Condition

Branch Condition: payload.output.CardType x String Equals

Each branch has a configured value:

Trigger Loopy demo Active Version 8 Last saved 12 days ago Save Delete Branch

Event Trigger Request

Branch Conditional My Condition

Supplier Payment

Subcontract PO

Purchase Requisition

Outgoing Claim

Adaptive Card Action My Action

Adaptive Card Action My Action

Adaptive Card Action My Action

Adaptive Card Action My Action

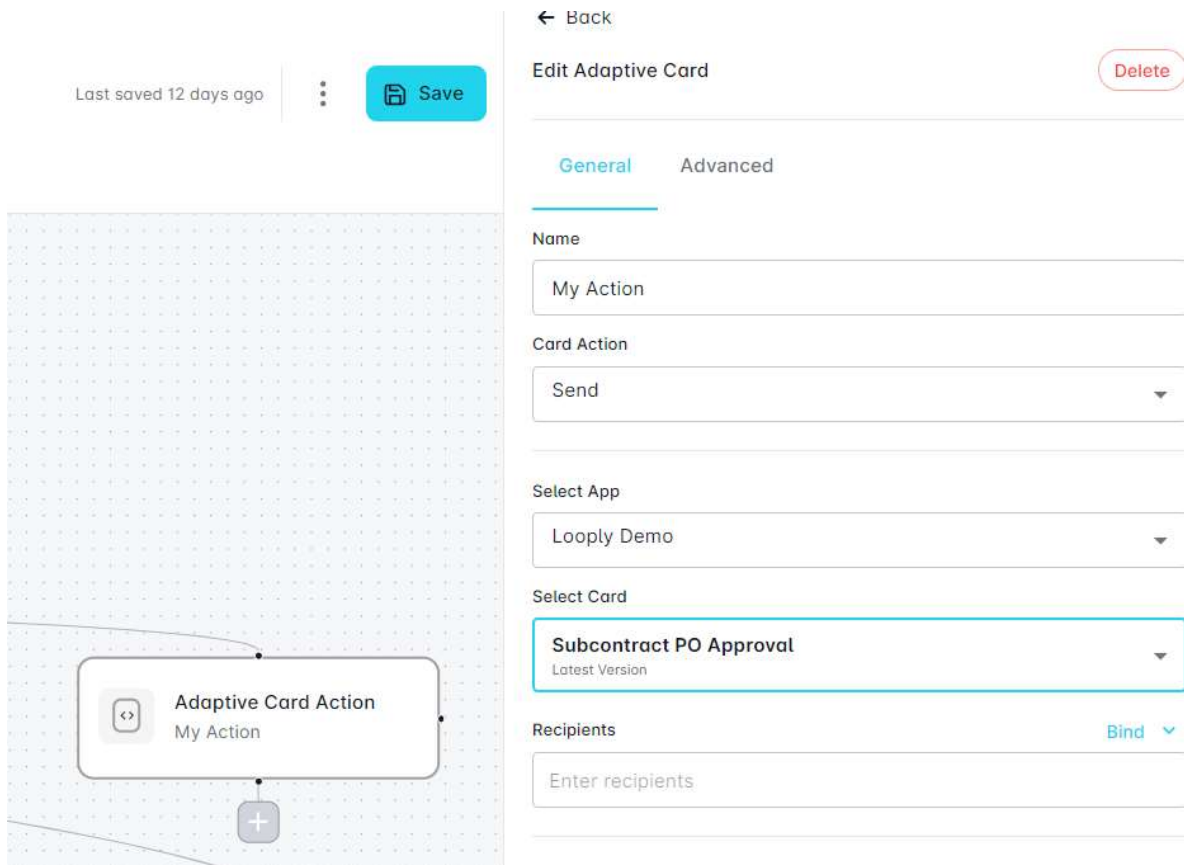
Condition Label: Supplier Payment

Condition Value: 1

Type: Default

At the end of each branch, I trigger a different adaptive card:

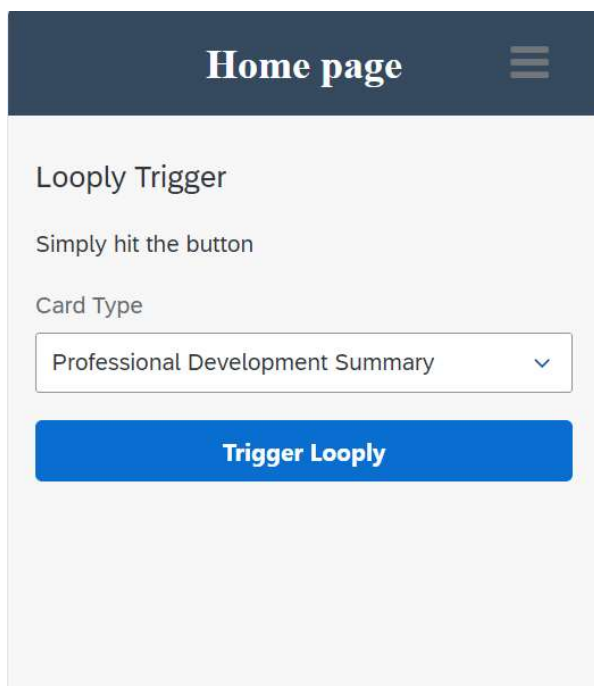
Linking a SAP Build app to Microsoft Teams



I select an app already created in Looply, and for the purpose of this demo, just hard-code a recipient email address.

Then I'm good to go to test. I can preview the app on desktop or phone:

Select a card type:



Card appears in Teams:

Linking a SAP Build app to Microsoft Teams

Professional Development Summary

Hi Chris,

Please be advised that following training is outstanding on your plan.

Security training courses are mandatory and must be completed by everyone in the company.

Course	Due date
Systems Security 2024	May 4, 2024
Harnessing AI in the office	July 31, 2024
Collaboration tools v2.5	July 31, 2024

Please contact HR Services if you have any questions.

[Confirm](#)

Summary

With no development I was able to build my demo, triggering different Adaptive Cards in Teams based on a simple SAP Build app.

video